Time-Encoded Values for Highly Efficient Stochastic Circuits

M. Hassan Najafi, *Student Member, IEEE,* Shiva Jamali-Zavareh, David. J. Lilja *Fellow, IEEE,* Marc D. Riedel, *Senior Member, IEEE,* Kia Bazargan, *Senior Member, IEEE,* and Ramesh Harjani, *Fellow, IEEE*

***Abstract*—Stochastic Computing (SC) is a promising technique for applications that require low area overhead and fault tolerance, but can tolerate relatively high latency. In the SC paradigm, logical computation is performed on randomized bit streams. In prior work, streams were generated with LFSRs; these contributed heavily to the hardware cost and consumed a significant amount of power. This paper introduces a new approach for encoding signal values: computation is performed on analog periodic pulse signals. Exploiting pulse width modulation (PWM), time-encoded signals corresponding to specific values are generated by adjusting the frequency and duty cycles of PWM signals. With this approach, the latency, area and energy consumption are all greatly reduced. Experimental results on image processing applications show up to 99% performance speedup, 98% saving in energy dissipation, and 40% area reduction compared to prior stochastic approaches. Circuits synthesized with the proposed approach can work as fast and energy-efficiently as a conventional binary design while retaining the fault-tolerance and low-cost advantages of conventional stochastic designs.**

***Index Terms*—Pulse Width Modulation, mixed-signal design, stochastic computing circuits, stochastic number generator, time-encoded values, energy-efficient computing.**

I. Introduction

Stochastic computing [1], [2] was originally advocated by Gaines in 1969 [3], and has gained traction in recent years again. It has been applied to a wide variety of applications such as image processing [4], [5], [6], [7], [8], error correction [9], [10], [11], and neural networks [12], [13], [14], [15], [16]. In SC, circuits operate on randomized bit streams. The signal value is encoded by the probability of obtaining a one versus a zero in the stream. In the "unipolar" representation, a real-valued number x $(0 \leq x \leq 1)$ is represented by a stream in which each bit has probability x of being one and probability $1-x$ of being zero. In the "bipolar" representation, a real-valued number y $(-1 \leq y \leq 1)$ is represented by a stream in which each bit has probability $\frac{y+1}{2}$ of being one and probability $1-\frac{y+1}{2}$ of being zero.

A stochastic representation is much less compact than conventional binary radix. However, complex operations can be performed with remarkably simple logic [17]. For example, a single AND performs multiplication with the unipolar representation; a single XNOR gate performs multiplication with the bipolar representation. A multiplexer implements scaled addition and subtraction. Complex functions, such as exponentials and trigonometric functions, can be computed through polynomial approximations.

M. H. Najafi, S. Jamali-Zavareh, D. Lilja, M. Riedel, K. Bazargan, and R. Harjani are with the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities, MN 55455 USA (e-mail: najaf011@umn.edu; jamal036@umn.edu; lilja@umn.edu; mriedel@umn.edu; kia@umn.edu; harjani@umn.edu).

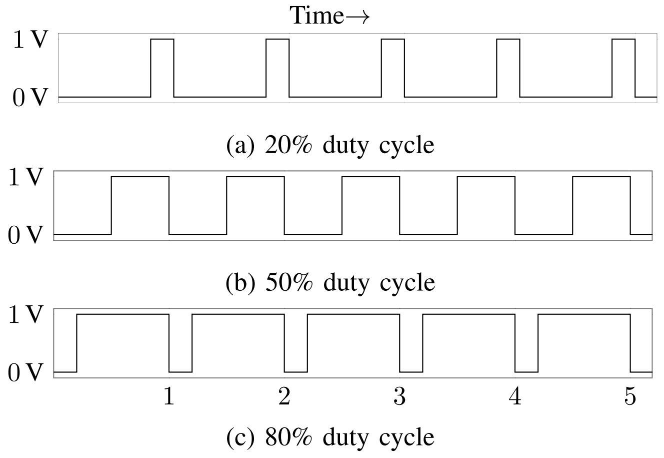

Fig. 1: PWM signals with different duty cycles.

In addition to producing simple and compact logic, a stochastic representation offers the advantage of error tolerance. In a noisy environment, bit flips will affect all the bits with equal probability. With a conventional binary radix representation, the high-order bits represent a large magnitude; accordingly, faults in these bits can produce large errors. In contrast, with a stochastic representation, all the bits are equally weighted. Hence, a single flip results in a small error. This error tolerance scales to high error rates so that multiple bit flips produce only small and uniform deviations from the nominal value.

A premise for SC is the availability of stochastic bit streams with the requisite probabilities. In prior work, these streams were generated from physical random sources [18][19] or with pseudo-random constructs such as LFSRs. These stochastic number generator (SNG) modules contributed heavily to the hardware cost. Indeed, in some cases, they accounted for 80% or more of the overall hardware cost [17]. Consequently, they also consumed a significant amount of power. Noting that $energy = power \times time$, the long run-time of stochastic circuits, together with the high power consumption of the SNGs, could lead to higher energy use than their conventional binary counterparts [20].

This paper introduces a new, energy-efficient, high-performance, and much less costly approach for generating stochastic bit streams using analog periodic pulse signals. Pulse width modulated (PWM) signals corresponding to specific values are generated by adjusting the frequency and duty cycles of PWM signals. The duty cycle $(0 \leq D \leq 1)$ describes the amount of time the signal is in the high (on) state as a percentage of the total time it takes to complete one cycle. As a result, the signal is encoded in *time*. The frequency $f = \frac{1}{T}$ of the PWM signal determines how long it takes to complete a cycle T and, therefore, how fast it switches between the high and the low states. Thus, a PWM signal $f(t)$ is defined as:

$$f(t) = \begin{cases} y_{\text{high}} & N \cdot T < t \leq N \cdot T + (1 - D) \cdot T \\ y_{\text{low}} & N \cdot T + (1 - D) \cdot T < t \leq (N + 1) \cdot T \end{cases}$$

where y_{high} and y_{low} are the high and low values of the signal, $N = 0, 1, 2, \cdots$ are the consecutive PWM cycles, and D is the duty cycle. Figure 1 shows three PWM signals with different duty cycles D when $T = 1$, $y_{\text{high}} = 1V$, and $y_{\text{low}} = 0V$.

Our approach is motivated by the following observation: a stochastic representation is a uniform, fractional representation. All that matters in terms of the value that is computed is the fraction of the time the signal is high [21]. For example, if a signal is high 25% of the time, it is evaluated as 0.25 in the unipolar format. Similarly, PWM signals can be treated as time-encoded inputs with values defined by their duty cycle. For example, the PWM signals shown in Figure 1 represent 0.2, 0.5, and 0.8 in the unipolar and -0.6, 0.0, and 0.6 in the bipolar representation.

The challenge is that PWM signals are not, in themselves, random or pseudo-random. Consider the stochastic operation of multiplication with a single AND gate. Taking the logical AND of bits in two independent bit streams yields the product of their probabilities, so an AND gate performs multiplication on stochastic bit streams. With PWM signals, we set the duty cycle to be the value represented. If two PWM signals have the same frequency, then the scheme will not work; the logical AND of the signals will not compute the product of the values. As we will show, the key is choosing *different* frequencies, and letting the system run over multiple PWM cycles. With the appropriate choice of frequencies for input signals, the high values intersect roughly as they would randomly. Thus, we achieve an inexpensive form of pseudo-randomness with PWM signals.

The paper is structured as follows. In Section II, we discuss prior approaches to stochastic number generation, introduce PWM signals as a representation, and demonstrate how to generate PWM signals. In Section III, we describe a methodology for performing SC on PWM signals. In Section IV, we validate the approach with experimental results. Finally, in Section V, we present conclusions.

II. Stochastic Number Generation

A. Conventional Stochastic Number Generators (SNGs)

Given an input value, say in binary radix, the conventional approach for generating a stochastic bit stream with probability x is as follows. Obtain an unbiased random value $0 \leq r \leq 1$ from a random [18][19] or pseudorandom source [22], [23]; compare it to the target value x; output a one if $r \leq x$ and a zero otherwise. Figure 2 illustrates the approach. The "random number generator" is usually an LFSR, which produces high quality pseudo-randomness [22]. In this approach, the period of the clock feeding the generator corresponds to the duration of a single bit in the output stream. Assuming that the pseudo-random numbers are uniformly distributed between $0 \ldots 2^M - 1$, the value stored in the constant number register should be $2^M \times x$. In the output, each bit is one with probability $(2^M \times x)/2^M = x$ [3][12].

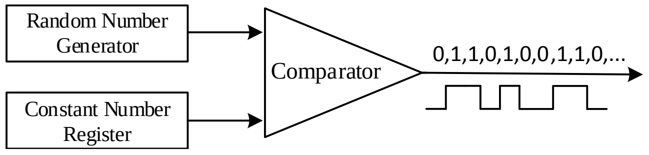

Fig. 2: Conventional Stochastic Number Generator.

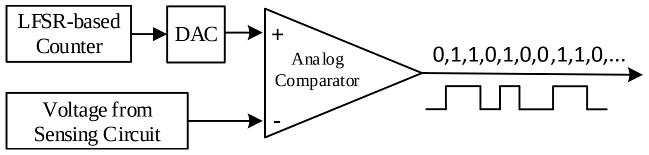

Fig. 3: SNG proposed in [6] for vision chips.

Pseudo-random number generators contribute heavily to the overall hardware cost. To represent real numbers with a resolution of 2^M, i.e. numbers of the form $\frac{a}{2^M}$ for integers a between 0 and 2^M, a stochastic representation requires a stream of 2^M bits. Generating streams with such resolution requires a generator that can produce 2^M unique values. Indeed, the high cost of the pseudo-random number generation diminishes one of the main advantages of SC: low hardware cost. Factoring in the cost of the generators, the overall hardware cost of an SC implementation is often comparable to that of a conventional representation [17].

Alaghi et al. [6] proposed a specific design of a SNG unit for vision chips. Vision chips have image sensors that convert the perceived light intensity to an analog electrical voltage. The sensed voltage is converted to a stochastic number by comparing it to a random voltage generated by an LFSR-based counter and a digital-to-analog converter (DAC). Figure 3 illustrates their approach. We will show that, by working with PWM signals, we can eliminate both the DAC as well as the LFSR. The result is a much less costly SNGs for applications that have analog electrical voltages as inputs.

B. PWM as the Stochastic Number Generator

In many electronic systems, existing analog inputs or on-board microcontrollers can be employed to generate PWM signals [25]. The simplest way to generate a PWM signal is to feed a sawtooth wave into the first input of an analog comparator and a control voltage into the second. The frequency of the sawtooth waveform determines the sampling rate of the signal. Thus, by changing the frequency of this wave, one can adjust the frequency of the generated PWM signal.

Figure 4 shows a common design for an analog PWM generator. The duty cycle of the PWM signal is set by changing the DC level of the input signal. The higher the DC level is, the wider the PWM pulses. The range of the DC signal varies between the minimum and maximum voltages of the triangle wave. For example, if we adjust the DC signal to have a level exactly half-way between the minimum and maximum, the circuit will generate a PWM signal with a duty cycle of 50%. This will correspond to an input value of 0.5 in the unipolar and 0.0 in the bipolar representation.

TABLE I: Area-Power comparison of different SNGs

SNGs	Unit	Area (μm^2)	Power @1-3GHz (μW)
Conventional SNG [17] (with 8-bit LFSR)	LFSR+Comparator	248	335-1013
	ADC [24]	>400,000	>10,000
Special SNG for vision chips [6]	LFSR	167	298-892
	DAC + Comparator	equivalent to an ADC	
PWM Generator (1-3 GHz freq.)	Comparator	20-58	65-192
	Ramp Generator	10-32	11-29
	Clock Generator	124-37	∼175

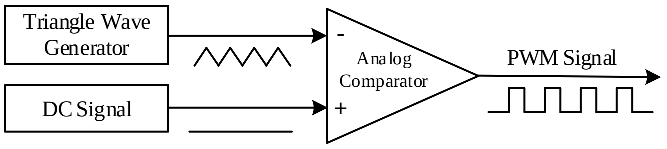

Fig. 4: A common analog PWM generator.

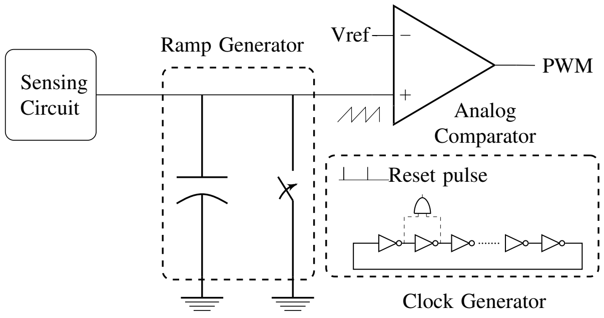

Fig. 5: The design of our PWM generator. The duty cycle is determined by the current coming from the sensing circuit (a photodiode, or a voltage controlled current source, etc) and the Reset pulse defines the frequency of the PWM signal. Vref is a fixed reference voltage.

Figure 5 shows the design of a low-cost PWM generator, consisting of a ramp generator, a clock signal generator, and an analog comparator. The input is a current coming from a sensing circuit that controls the duty cycle of the PWM signal. The clock generator provides the required *Reset* signal which determines the frequency of the PWM signal. Ring oscillators consisting of an odd number of inverter gates can be used as the clock generator. The frequency of the *Reset* clock can be adjusted by either changing the supply voltage or changing the number of inverters in the oscillator. In the 45nm technology, a ring of approximately 89 inverter gates can generate a local clock with a period of 1ns with a supply voltage of 1.0V.

Table I shows an area-power comparison of the proposed PWM generator shown in Figure 5 with prior methods for SNGs: 1) the LFSR-based method in [17], and 2) the method proposed for vision chips in [6]. The results are for 45nm technology. We assume that the inputs are analog voltages or currents coming from a sensing circuit. The effective number of bits (ENOB) corresponding to different frequencies of the PWM generator is shown in Figure 6. Analog-to-digital convertors (ADCs) are used to obtain a digital representation for the LFSR-based method. The cost of a 45nm SAR ADC is taken from [24]. The special SNG proposed for vision chips resembles an ADC; we assume that it is roughly as expensive as a SAR ADC. The Synopsys Design Compiler was used to synthesize the SNGs. The results in Table I demonstrate that our mixed-signal method, based on PWM generators, has much lower area and power costs than the prior methods in cases where the inputs are in analog voltage or current form. Accordingly, the approach is a good fit for real-time image processing circuits, such as those in vision chips. These have image sensors that convert the perceived light intensity to an analog voltage or current.

Note that in prior methods a counter was used to convert stochastic streams back into real values in the digital domain. To convert the stochastic signals directly to a value in the analog domain, prior work used a simple RC integrator circuit to average the signal [26], [27]. For a faster response time, we use a Gm-C active integrator to average the output from processing PWM signals and measure the fraction of the time that the signal is high. For example, for a PWM signal with a period of T, duty cycle of D, $y_{\text{high}} = 1V$, and $y_{\text{low}} = 0V$, the integrator gives the

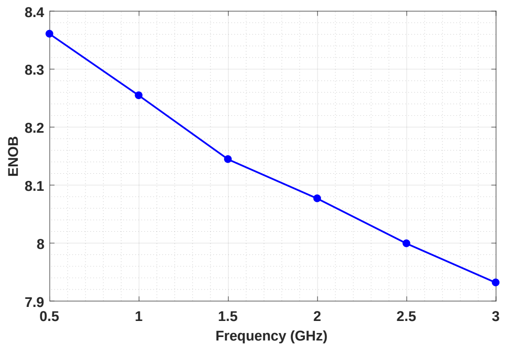

Fig. 6: The ENOB of the proposed PWM generator shown in Figure 5 when generating PWM signals with frequencies from 0.5 to 3 GHz. More detail on the noise modeling of the implemented PWM generator will be discussed in Section V.

average value of the first period of the signal as follow:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)dt = \frac{1}{T}\left(\int_0^{(1-D)T} y_{\text{low}}dt + \int_{(1-D)T}^{T} y_{\text{high}}dt\right)$$
$$= \frac{1}{T}(T \cdot (1-D) \cdot y_{\text{low}} + T \cdot D \cdot y_{\text{high}}) = D$$

III. Stochastic Systems with PWM Signals

In this section, we first discuss the implementation of basic stochastic operations operating with PWM signals. Then we extend the discussion to more complex examples consisting of a multi-level combination of stochastic operations.

A. Stochastic Operations with PWM signals

1) Multiplication: In the SC representation, a single AND (XNOR) gate performs multiplication if the unipolar (bipolar) format is used. The multiciplication operation presumes that the

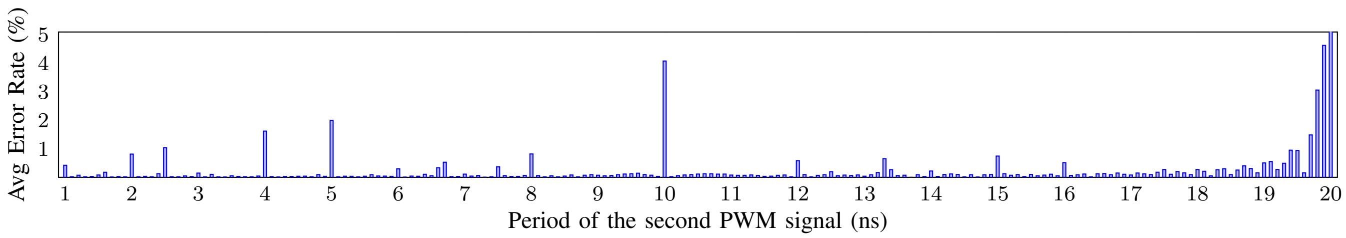

Fig. 7: The average error rates when performing a multiplication operation using an AND gate for 1000ns on 1000 sets of random input values when the inputs are represented using PWM signals. The period of the first input is set at 20ns while the period of the second changes from 1ns to 20ns.

inputs are independent, uncorrelated streams [12]. Connecting two PWM signals with the same duty cycle and the same frequency to the inputs of an AND gate will evidently not work. It produces an output signal equal to the two inputs, not the square of the value as required. However, as we will show, one can use PWM signals provided that they have different frequencies (recall that we represent values by the duty cycle of PWM signals, not their frequency).

Instead of continuous-valued time signals, assume for the sake of argument that PWM signals are represented as bit streams. For instance, assume an input value $X = 3/5$ (a signal with duty cycle of 60%) is represented by the bit stream 11100, and an input $Y = 1/2$ (a duty cycle of 50%) is represented by the bit stream 1100. Note that the stream for X has length 5 while that for Y has length 4. Suppose we multiply X and Y with an AND gate. Let the bit streams run for 20 clock cycles, corresponding to 4 repetitions of X and 5 repetitions of Y. Taking the bit-wise AND of the streams

$$\begin{array}{r l} \text{X} & = 11100111001110011100 \\ \text{Y} & = 11001100110011001100 \\ \hline \text{X.Y} & = 11000100000010001100 \end{array}$$

we observe 6/20 ones in the output, the expected value, since $3/5 \times 1/2 = 6/20$. The results of this sort of multiplication operation is always correct if one chooses stream lengths that are *relatively prime* and let them run up to *the common multiple*. This is because when the length of the inputs are relatively prime the difference between the lengths results in a new phase between the signals in each repetition until they get to the common multiple. A new initial phase in each repetition causes each bit of the first bit stream to see every bit of the second stream. This is, intuitively, equivalent to sliding one bit stream past the other. The bit streams are therefore multiplied by convolving through sliding and ANDing repeatedly [28], [29].

Proof. Let a/m be represented by a stream of m bits consisting of a bits of 1's with the rest of the bits being 0. Similarly, let b/n be represented by a stream of n bits with b bits of 1's and the remaining bits being 0. Assume that we repeat both streams to reach a total of the least common multiple (LCM) number of bits, or for simplicity mn bits in each stream. Applying an AND gate to these streams, we will have $a \times b$ bits of 1's if and only if the set $\{mk + i \pmod{n} : k = 0, 1, \ldots, n-1\}$ is a complete set of residues mod n. Here, i is the position of any 1 bit in the first stream. The first observation is that, whether the above holds or not does not depend on i. The second observation is that, when $i = 0$, this statement is true if and only if m and n are relatively prime. Therefore, ANDing the above streams produces $\frac{a}{m} \times \frac{b}{n}$ if and only if m and n are relatively prime. Q.E.D.

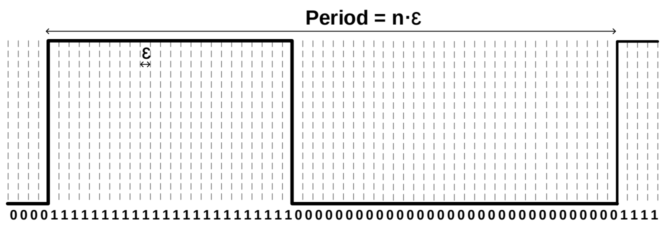

Fig. 8: Discretizing a continuous PWM signal.

This argument can be easily expanded to analog PWM signals if the continuous signals are discretized into bit streams, as illustrated in Figure 8. A PWM signal can be discretized into a bit stream by dividing the signal into pulses of size $epsilon$ and assigning 0/1 bits to these pulses. The relatively prime length rule is then applicable to this discrete representation of the PWM signals and continues to hold as $\epsilon \to 0$. Note that in signal processing terminology, PWM signals with relatively prime periods are inharmonic.

To illustrate this argument, we simulated multiplication on a thousand sets of random input values represented by PWM signals in Matlab [30]. We fixed the period of the first PWM signal at 20ns while varying the period of the second from 1ns to 20ns in increments of 0.1ns. For each pair of periods, we converted the randomly generated sets into corresponding PWM signals and then performed multiplication for 1000ns. The accuracy of the results was verified by calculating the difference between the expected value and the measured output value for all sets. To convert the output signals into deterministic real values, we measured the fraction of the time that the output is high and divided this by the total time. The average error rates for multiplication for different pairs of periods are shown in Figure 7.

As can be seen in Figure 7, with the period of the first PWM input signal fixed at 20ns, choosing 1ns, 2ns, 2.5ns, 4ns, 5ns, 8ns, 10ns, 12ns, 15ns, 16ns, or values very close to 20ns as the period of the second PWM input signal produces poor results. This can be attributed to an aliasing effect that occurs with periodic signals that are harmonically related. Eliminating these choices, the measured average error rate for other values was always less than 0.5%. Note that these results could ideally be extended to any other range of periods[1]. For example, knowing that 20ns and 13ns is a good pair, periods of 2ns and 1.3ns, or 10ns and 6.5ns would work equally well. From this observation we make our first conclusion:

[1]In practice, the resolution or effective number of bits (ENOB) of the PWM signals can affect the accuracy and so limits the extension range.

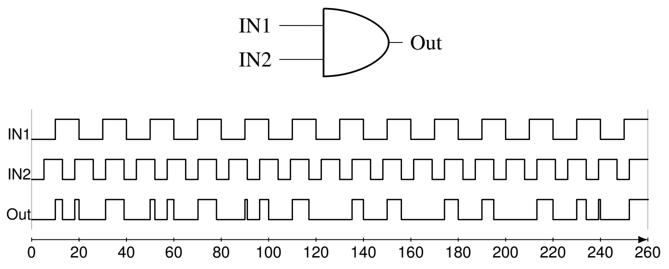

Fig. 9: An example of multiplying two PWM signals using an AND gate. IN1 represents 0.5 (50% duty cycle) with a period of 20ns, and IN2 represents 0.6 (60% duty cycle) with a period of 13ns. The output signal from t=0ns to 260ns represents 0.30 (78ns/260ns=3/10), the expected value from multiplication of the inputs.

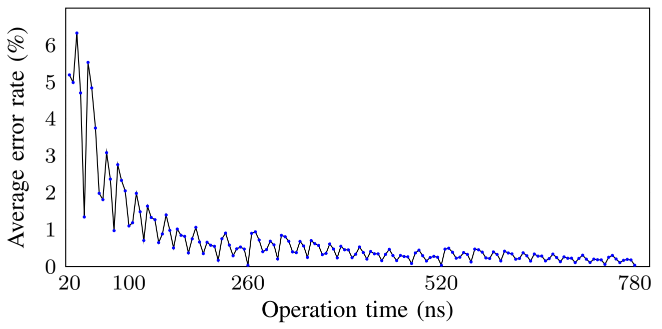

Fig. 10: The average error rate of multiplying 1000 pairs of random numbers represented by PWM signals when varying the operation time. The period of the PWM signals corresponding to the first and to the second number in each trial is 20ns and 13ns, receptively.

Conclusion 1. Stochastic multiplication of numbers represented by PWM signals produces highly accurate results if the signals are not harmonically related.

With inharmonic PWM signals as inputs of multiplication, the fraction of time that the output signal is high will converge to the expected value eventually. However, stochastic circuits are not energy-efficient if the operations run more than what they actually need to. The question is: How many cycles of PWM signals are required to reach to a reasonable accuracy? Figure 9 shows an example of multiplying two stochastic numbers, 0.5 and 0.6, represented using two PWM signals. The period of the first PWM signal is 20ns and that of the second is 13ns. The figure shows that, after performing the operation for 260ns, the fraction of the total time the output signal is high equals the value expected, when multiplying the two input values, namely 0.3.

Expanding the example above to different operation times, Figure 10 shows the average error rates of multiplying 1000 pairs of random numbers represented by PWM signals when a fixed period of 20ns is selected for the first and a fixed period of 13ns is chosen for the second. We vary the operation time. As the figure shows, the output of multiplications converges to the expected value if the operations continue at least up to the least common multiple (LCM) of the periods of the input signals (here, $20 \times 13 = 260ns$). The best possible accuracy is obtained when the operation is run for exactly the LCM (260ns) or multiples of the LCM (520ns and 780ns). Running the operation longer than the LCM does not help the accuracy. This is in contrast to prior SC approaches where increasing the length of bit streams improves the quality of the results.

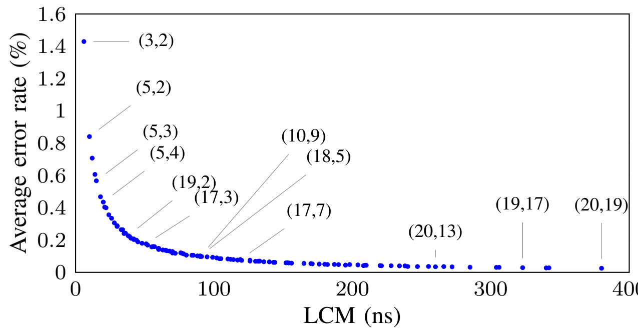

Fig. 11: The average error rate for multiplying 1000 pairs of random numbers represented by PWM signals when the period of the first and the second PWM signal are relatively prime integers in the interval [2, 20]. A lower average error rate in the figure means a higher ENOB in the computations.

Let us consider the X.Y stream produced before. The LCM of the input streams was $4 \times 5 = 20$ and after exactly 20 cycles the expected output was produced. Continuing the operation for another 20 cycles produces exactly the same output with the same ratio of ones to the length of stream:

$$\begin{array}{rl} X & = 11100111001110011100\ 11100111001110011100 \\ Y & = 11001100110011001100\ 11001100110011001100 \\ \hline X.Y & = 11000100000010001100\ 11000100000010001100 \end{array}$$

Thus, we can say that the output has a period of 20 cycles. A similar result is observed when ANDing continuous PWM signals. The output has a period of the LCM. The signal produced from the first LCM to the second LCM is exactly the same as the signal produced from time=0 to the first LCM. This motivates our second conclusion:

Conclusion 2. The best accuracy when multiplying numbers represented by PWM signals is obtained when running the operation for the LCM or multiples of the LCM of the period of the inputs.

Knowing that relatively prime periods must be selected for the input signals and the multiplication operation should be run for the LCM of the periods, a new question arises:

Considering available sets of relatively prime periods, each with a different LCM, what is the best set of periods to reach to a desired accuracy? For example, (17ns, 3ns) and (17ns, 7ns) are two possible sets of periods to generate the PWM input signals for a multiplication operation. The first set has an LCM of 51ns while the second's is 119ns. Which one of these two sets is a better choice?

Figure 11 shows the average error rates of multiplying 1000 pairs of random numbers represented by PWM signals when different sets of relatively prime periods are selected as the periods of the input signals and the operations are run for the LCM of the periods. Each set of periods has a different LCM. As can be seen in the figure, the larger the LCM, the lower the average error rate. The reason is that larger LCMs are produced by longer periods and a longer period means a higher ENOB in representing the input values and so a higher ENOB in the computations. Note that while generating PWM signals with longer periods and so larger LCMs gives more accurate results, this requires a longer operation time. Thus, if a set of periods with a smaller LCM can satisfy the accuracy requirements, this might be the better choice. Thus, we conclude the following:

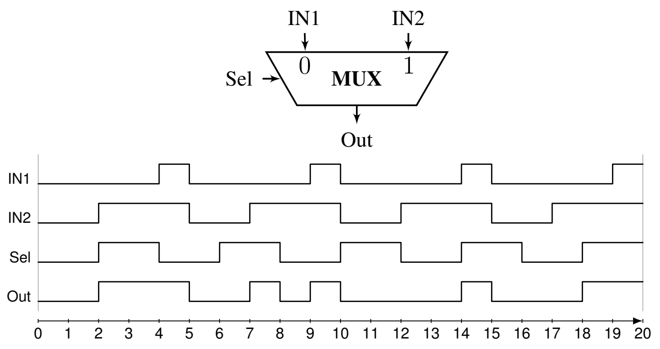

Fig. 12: An example of the scaled addition of two PWM signals using a MUX. IN1 and IN2 represent 0.2 and 0.6 with a period of 5ns, and Sel represents 0.5 with a period of 4ns. The output signal from t=0ns to 20ns represents 0.40 (8ns/20ns=4/10), the expected value from the scaled addition of the inputs.

Conclusion 3. The larger the LCM of the periods of the PWM input signals, the higher the accuracy when performing multiplication.

Note that, although we based our argument on multiplication of stochastic numbers in the unipolar encoding, the conclusions are also applicable to the bipolar format. The AND gate is replaced by an XNOR gate while the input signals are still required to not be harmonically related and the operation must continue for an LCM number of periods.

2) Scaled Addition and Subtraction: Stochastic values are restricted to the interval [0, 1] (in the unipolar case) or the interval [-1, 1] (in the bipolar case). So one cannot perform addition or subtraction directly, since the result might lie outside these intervals. However, one can perform scaled addition and subtraction. These operations can be performed with a multiplexer (MUX) [4]. The performance of a MUX as a stochastic scaled adder/subtracter is insensitive to the correlation between its inputs. This is because only one input is connected to the output at a time [31]. Thus, highly overlapped inputs like PWM signals with the same frequency can be connected to the inputs of a MUX. The important point when performing scaled addition and subtraction with a MUX on PWM signals is that the period of the select signal should not be harmonically related to the period of the input signals. For example, 5ns, 5ns, and 4ns is a good set of numbers for the period of the first, the second, and the select input signals, respectively.

Figure 12 shows an example of scaled addition on two stochastic numbers, 0.2 and 0.6, represented by two PWM signals (both have periods of 5ns). A PWM signal with duty cycle of 50% and period of 4ns is connected to the select input of the MUX. As shown, after performing the operation for 20ns, the fraction of the total time the output signal is high equals the expected value, 0.40. The same argument we had for the multiplication operation also exists here – the scaled addition/subtraction operation should be run for the LCM or multiples of the LCM of the period of the input signals and that of the select signal to produce the correct output. Note that choosing different periods for the main inputs of the MUX results in a larger LCM and so results in a longer operation time. Furthermore, generating inputs with different periods requires extra clock generator circuitry. We conclude that it is most efficient to generate signals for the main inputs of the MUX with the same period.

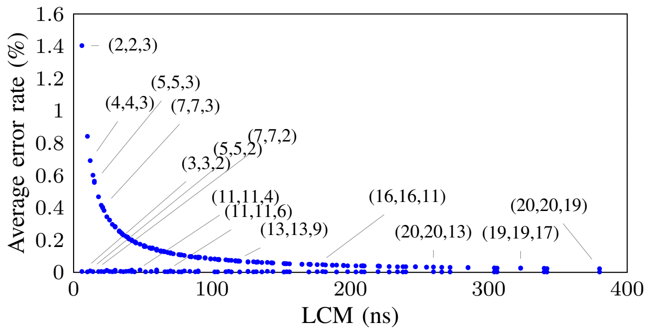

Fig. 13: The average error rate of performing scaled addition on 1000 pairs of random numbers represented by PWM signals when the period of the first and the second PWM signal are the same but different and relatively prime with the period of the PWM select signal. The periods are selected from integers in [2, 20] interval.

A unique property of MUX-based operations is that large LCMs are not necessarily required to produce accurate results. Similar to what we saw for the multiplication operation, selecting inharmonic periods with a large LCM guarantees the accuracy of the results for the scaled addition/subtraction. However, it is possible for the stochastic MUX-based operations to produce accurate results even with inputs with very small periods. Figure 13 shows the average error rate of performing scaled addition when inharmonic PWM signals are connected to the main and select inputs of the MUX. Each point in the figure represents the accuracy and the LCM corresponding to one set of periods. The first and the second numbers in each set are the period of the main PWM inputs and the third number is the period of the select input. As the results show, when the period of the PWM select signal is an "even" value (2ns, 4ns,...) choosing "odd" periods as the period of the main PWM inputs result in highly accurate outputs. When choosing an "even" period for the inputs and an "odd" period for the select signal, a large LCM is needed to produce accurate results. The reason is shown in Figure 14. A select signal with an "even" period perfectly splits an input with an "odd" period in two periodic parts with the same duration at the high level. Thus, it does not matter to which input of the MUX the input signal is connected. However, in the case of an "odd" period for the select signal, connecting the input signal to different inputs of the MUX selects different parts of the input signal with different high durations. This motivates our fourth conclusion:

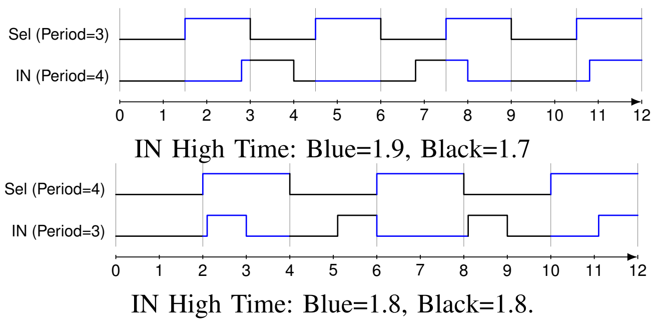
Sel (Period=3)
IN (Period=4)
IN High Time: Blue=1.9, Black=1.7
Sel (Period=4)
IN (Period=3)
IN High Time: Blue=1.8, Black=1.8.

Fig. 14: Examples of choosing an "odd" or an "even" number as the period of the MUX's select signal. The input is a PWM signal with D=30%. Black (blue) lines are parts of the input signal that will be connected to the output of the MUX when the input is connected to the first (second) input.

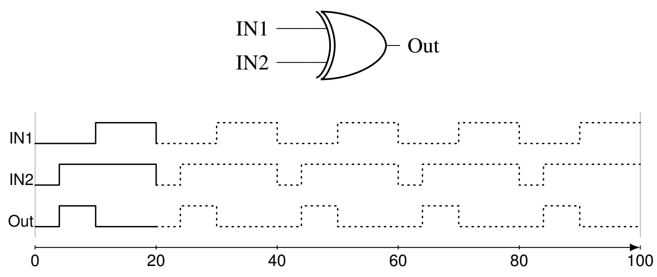

Fig. 15: An example of performing stochastic absolute value subtraction using an XOR gate when two synchronized PWM signals are used as the inputs of the gate, one representing 0.5 (D=50%) and the other 0.8 (D=80%). Both PWM signals have a period of 20ns. The output signal from t=0ns to 20ns represents 0.3, the expected value for $|IN1 - IN2|$.

Conclusion 4. Optimal choices for MUX-based operations are those with an "even" value for the period of the select input and an "odd" value for the period of the main inputs. The operation should run for the LCM of the periods.

3) Absolute Value subtraction: Correlation between the inputs of a stochastic circuit can sometimes change the functionality of a circuit, which might result in a more desirable operation. An XOR gate with independent inputs performs the function $z = x_1(1-x_2)+x_2(1-x_1)$. However, when fed with correlated inputs where the two input streams have maximum overlap in their 1s, the circuit computes $|x_1-x_2|$ [6]. Consider $x_1 = 11101$ and $x_2 = 10001$, two 5-bit long correlated stochastic streams representing 4/5 and 2/5. Connecting these streams to the inputs of an XOR gate produces $Y = 01100$, the expected value from performing absolute valued subtraction. In this case, the output stream has the same number of bits as the input streams.

When working with PWM signals, high correlation or maximum overlap is provided by satisfying two requirements: 1) choosing the same frequency for the input signals, and 2) having maximum overlap between the high parts of the signals. Thus, two PWM signals that have the same period, with the high part in each one located at the start or end of each period, are called correlated (or synchronized) signals [32]. Figure 15 shows an example of performing absolute value subtraction on two synchronized PWM signals. As the figure shows, the correct output with the highest possible accuracy is ready right after performing the operation for only one period of the PWM input signals. Thus:

Conclusion 5. For operations like absolute value subtraction which work only with correlated inputs (synchronized PWM signals), the period of the output signal, and, thus, the operation time, equals the period of the input signals.

This conclusion introduces an important advantage of working on the synchronized PWM signals which is that they eliminate the requirement of running the operation for several repetitions of the input signals to obtain an accurate output signal. An important point, however, is that there is a limitation in using such operations that require highly correlated inputs. Providing the required synchronization (maximum high part overlap between the input signals) is difficult for the second (or higher) level of the circuit where the signals are the outputs of a previous level. Nonetheless, performing these operations can still be advantageous at the first level of circuits.

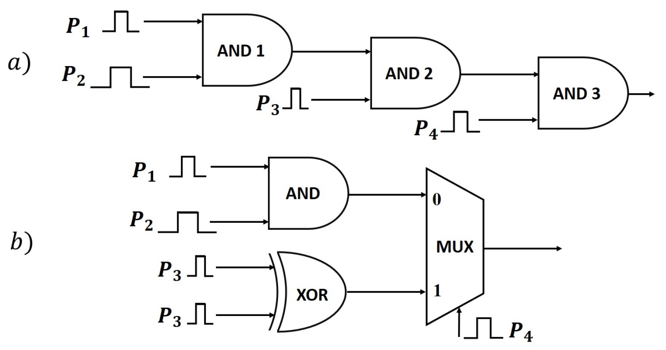

Fig. 16: Examples of multi-level stochastic circuits.

B. Multilevel Circuit PWM signals

In the following we briefly discuss the functionality of multi-level stochastic logic when PWM signals are used as the inputs of the circuit. An interesting point in performing stochastic operations on PWM signals is that the output of each level can be used as the input of the next level even though the output is not a PWM signal. When connecting two PWM signals to a stochastic operator, the output is a conventional stochastic number whose value cannot be found from the duty cycle but rather by probability of being in the "high" state. However, the main difference between such an output with a conventional random stochastic signal is that, since the primary inputs were PWM signals, the generated output is a periodic signal. This property allows us to use the output of each level as the input to the next level. By knowing the period of the output signal, the obtained signal and some new signals that are not harmonically related can be used in the subsequent levels.

Consider the example presented in Figure 16.a, a 3-level circuit multiplying four PWM signals with periods of $P1, P2, P3$ and $P4$. We want to choose the periods of the inputs and the required operation time which can lead to accurate outputs. Based on the conclusions in Section 2, $P1$ and $P2$ should not be harmonically related. The AND1 gate should operate for $i \times P1 \times P2$ (i is an integer ≥ 1). The output of the AND1 gate is a signal with a period of $P1 \times P2$. The accuracy of the output produced by AND2 depends on the output of AND1 and also on $P3$, the period of the third PWM signal. $P3$ should not be harmonically related to the period of the signal generated at the output of AND1, and so to $P1$ and $P2$. Finally, P4 should not be harmonically related to $P1$, $P2$, and $P3$. The final output has a period of $P1 \times P2 \times P3 \times P4$, so the circuit must run for this amount of time to produce an accurate result.

Expanding the example above to circuits multiplying N PWM signals with N periods that are not harmonically related, the operation time must be the LCM of *all* these periods. The important trade-off here is to select small or large periods for these signals. Small periods results in a small LCM, and so need a shorter operation time. Larger periods have larger LCMs and so require a longer running time. As shown in Figure 11, the larger the LCM, the higher the accuracy of multiplication. Thus, selecting the period of the PWM signals for such circuits depends on the accuracy and timing expectations.

The circuit presented in Figure 16.b incorporates all three sorts of basic operations. The AND gate's output has a period of $P1 \times P2$ while the output of the XOR gate has a period equal

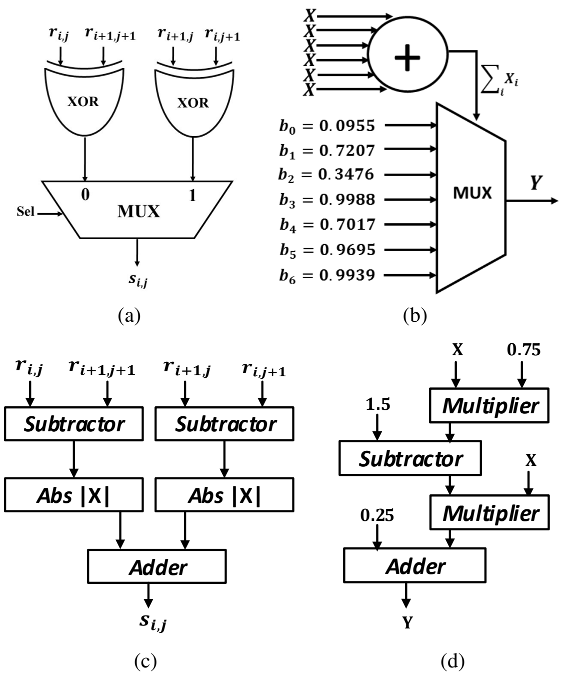

Fig. 17: Robert's cross edge detection algorithm a) core stochastic logic [6], c) conventional binary implementation. Gamma correction function b) core stochastic logic based on ReSC architecture [17], d) a conventional binary implementation [6].

to the period of its inputs, or $P3$. The minimum operation time for this circuit is obtained when the MUX's inputs have the same periods ($P1 \times P2 = P3$). $P3$ must be an "odd" number while a small even value must be selected for P4. For this circuit the total operation time will be $P3 \times P4$. In cases where $P3 \neq P1 \times P2$, the total operation time will be the LCM of the period of all inputs, or $P1 \times P2 \times P3 \times P4$.

IV. Experimental Results

To validate our ideas, we used stochastic implementations of two well-known digital image processing algorithms, the Robert's cross edge detection algorithm and the Gamma correction function. The core stochastic computation circuit for the Robert's cross algorithm was taken from [6], and the core logic for the gamma correction algorithm was taken from [17] (both shown in Figure 17). In the rest of this section, when we refer to the "prior" approach, we pair the core stochastic logic with input SNGs (LFSR+comparator as shown in Figure 2), and output counters to convert stochastic bitstreams to binary numbers. When we refer to the "PWM" approach, we pair the core stochastic logic with PWM generators (Figure 4) and a voltage integrator to generate the analog output. The conventional binary implementations of the selected algorithms are also shown in Figure 17.

We implemented SPICE netlists for the stochastic circuits described above. Two 128×128 sample images (16384 pixels each) were selected for the simulations. Simulations were carried out using a 45-nm gate library in HSPICE. We implemented the PWM generator proposed in Figure 5 for converting input pixel values into the corresponding PWM signals. Figure 18 shows the input sample images as well as the output of processing these images using a deterministic, software-based implementation of each algorithm in Matlab. We call this the "golden" approach, with a 0% average error rate. Also, we simulated the circuit operation on randomized stochastic streams in the "prior" approach. The conventional SNG described in Figure 2 was used for converting input pixel intensities into stochastic bit streams. An 8-bit maximal period LFSR was used as the pseudo-random number generator. Bit streams 256 bits long were generated for each input value. We calculate the average output error rate for the output image produced by the "prior" approach as follows:

$$E = \frac{\sum_{i=1}^{128}\sum_{j=1}^{128} |T_{i,j} - S_{i,j}|}{255 \cdot (128 \times 128)} \times 100$$

where $S_{i,j}$ is the expected pixel value in the output image and $T_{i,j}$ is the pixel value produced using the circuit.

To compare the operation time of the PWM approach with the delay of the prior approach, and also that of the conventional binary approach, we synthesized the Robert's cross and the gamma correction circuits using the Synopsys Design Compiler vH2013.12 [33] with a 45nm gate library. The stochastic circuits had a critical path of 0.34ns and 0.60ns, respectively. In the following subsections we first describe the process of synthesizing the selected circuits with the proposed PWM approach and then compare performance, area, and energy dissipation of the implemented circuits.

A. Case Study 1: Robert's cross edge decetor

Each Robert's cross operator consists of a pair of 2 × 2 convolution kernels that process an image pixel based on its three neighbors as follows:

$$S_{i,j} = \frac{1}{2} \times (|r_{i,j} - r_{i+1,j+1}| + |r_{i,j+1} - r_{i+1,j}|)$$

where $r_{i,j}$ is the value of the pixel at location (i, j) of the original input image and $S_{i,j}$ is the output value computed for the same location in the output image. Figure 17.a shows the stochastic implementation of the Robert's cross algorithm proposed by Alaghi and Hayes [6], consisting of a MUX for the scaled addition and two XOR gates to perform the absolute value subtractions. This circuit is the core computation logic and is shared between the "prior" stochastic approach and our PWM approach.

1) Prior Method [6]: To generate the circuit for the prior approach, we pair the core stochastic logic of Figure 17.a with one LFSR and four comparators to generate the input streams feeding the XOR gates. Only one LFSR is used for the XOR input lines because Alaghi's approach relies on *correlated* bitstreams. Another LFSR and comparator is also necessary to generate the select stream. Note that when the input is given in analog voltage, coming from a sensing circuit, an ADC must also be used to convert the analog input signal into digital form. We ignore the ADC unit in our comparisons. If the cost of the ADC were to be added, our approach would have shown even larger gains compared to prior work. The output of the prior approach circuit is fed to a counter to convert the bitstream to a binary number.

2) The PWM Method: Next we describe how we implemented the Robert's cross algorithm using the PWM approach. The core stochastic logic of Figure 17.a is paired with PWM generators that provide the input signals feeding the XOR gates, and the output of the MUX is fed to a voltage integrator circuit. The following steps are used to synthesize the circuit in the PWM approach:

Step 1. Frequency Selection. When using PWM signals as inputs of a stochastic circuit, one has to select appropriate frequencies. As discussed in Section III-A3, the inputs to an XOR gate must be two synchronized PWM signals to compute the absolute value subtraction. Since the MUX unit is also insensitive to the correlation between input signals, four synchronized PWM signals corresponding to four pixels of the image can be connected to the main inputs of the Robert's cross circuit. The important point here is to appropriately select the frequency of the PWM signal connected to the select line of the MUX. This select signal can be a clock signal which is a PWM signal with 50% duty cyle. The period of this signal must not be harmonically related to the period of the main inputs of the MUX. Since the period of the signal produced at the output of the XOR gates is the same as the period of their inputs, the period of the clock signal must not be harmonically related to the period of the circuit's main inputs. Considering the critical path (0.34ns) as the minimum allowed period of the PWM signals, we chose 0.51ns as the period of the main PWM input signals and 0.34ns as the period of the select signal. These numbers are obtained by scaling (3ns, 2ns) down which is one of the best set of periods extracted in Section III-A2.

Step 2. Operation Time Determination. We showed that the results of performing stochastic absolute value subtraction is ready after running the operation for only one period of the input PWM signals. For scaled addition/subtraction operations, the best operation time is the LCM of the periods of the MUX select and input signals. Since we scaled (3ns, 2ns) down to (0.51ns, 0.34ns), the best operation time is also obtained by scaling their LCM down by the same scaling factor. Thus, the best operation time for the synthesized Robert's cross circuit in the PWM approach is 1.02ns.

Step 3. Clock Generation. Since the frequency of all four PWM inputs is the same, a clock generator with an oscillation period of 0.51ns is enough to drive the main PWM generators. A second clock signal with a period of 0.34 is also necessary for the select line of the MUX. Thus, a total of two clock generators would be sufficient for generating the inputs of the Robert's cross circuit. We used rings of 43 and 29 inverters to generate the required clock signals.

3) Comparison: We processed each image pixel separately and computed the corresponding output value. Comparing the produced output image in the PWM approach with the golden image, the mean of the output error rates was 1.28%. Thus, the proposed approach could decrease the average error rate of processing the sample image when it is compared with that of the prior stochastic approach with 256-bit streams (1.49%). Considering the delay of the prior stochastic approach ($256 \times 0.34ns = 87.04ns$), the PWM approach decreases the processing time of each pixel by more than 98%, to only 1.02ns. Even if one argues that the quality of the 32-bit streams (1.98%) is enough for the prior approach, still the PWM approach has improved the operation time by 90%. The area, power, and energy consumption of the circuit when working with PWM signals are also presented and compared with the prior approach in Table II. From the area, area-delay and energy numbers, we see that the proposed PWM approach has a significant cost advantage when compared with the prior stochastic approach.

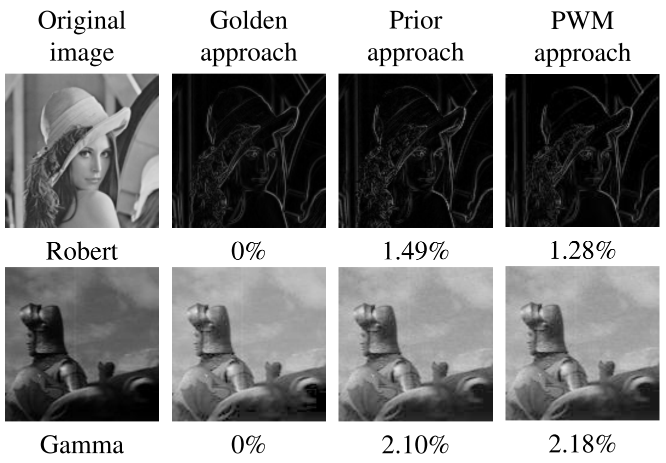

Fig. 18: The original 128*128 sample images and the outputs of processing the input images using the "golden approach", the "prior approach", and the proposed PWM approach with the Robert's cross stochastic circuit (first row) and the Gamma correction stochastic circuit (second row).

Compared to the conventional binary implementation, although the PWM approach is slightly slower, it costs 63 percent less area, dissipates 12 percent less energy, and reduces the area-delay product by more than 50 percent. The main barrier to practical use of the prior stochastic implementation was its long latency and correspondingly high energy use. However, as the results presented in Table II show, the proposed PWM approach is able to implement the Robert's cross edge detection algorithm with the advantages of the stochastic design but as fast and energy-efficiently as the conventional binary design.

B. Case Study 2: Gamma Correction

A flexible and straight-forward method to utilize SC in different applications is to synthesize the SC circuits with a MUX-based architecture, called ReSC [17]. This design approach is simple and area-efficient, and is able to realize polynomial functions that can be translated to Bernstein Polynomials. The gamma correction function ($f(x) = x^{\gamma}$) is a popular pixel value transformation that can change luminance and tri-stimulus values in video and image processing systems. This function can be approximated using a Bernstein polynomial. A stochastic implementation of the gamma correction function for $\gamma = 0.45$ based on the ReSC architecture is shown in Fig. 17.b. The inputs to this system consist of six independent bit streams, each with a probability corresponding to the value x of the input pixel (denoted as x in the figure), as well as seven random bit streams set to constant values, corresponding to the Bernstein coefficients, $b_0 = 0.0955$, $b_1 = 0.7207$, $b_2 = 0.3476$, $b_3 = 0.9988$, $b_4 = 0.7017$, $b_5 = 0.9695$ and $b_6 = 0.9939$. Additional details of the circuit can be found in [17].

In the following subsections, just as we did in Case Study 1, we use the same core stochastic logic for the prior and the PWM methods, but use different input SNG and output accumulation circuits.

1) Prior Method [17]: Based on the analysis done in [34], we can use delayed outputs of the same bitstream to generate

TABLE II: Area, delay, power and energy comparison of the implemented circuits for the conventional binary, prior stochastic and the proposed PWM approach. For the prior stochastic approach we ignore the cost of the ADC. Delay and power numbers are reported for the maximum working frequency.

Circuit	Approach	Area (μm^2)				Delay (ns)	Power (μW) (@max freq.)	Energy (pJ)	Area$\times$Delay ($\mu m^2 \times \mu s$)
		Core	SNG	Output Circt.	Total				
Robert	Conventional binary	1626	-	-	1626	0.78	1415	1.10	1.26
	Stochastic-Prior	22	739	199	960	87.04	2813	244.8	83.55
	Stochastic-PWM	22	464	110	596	1.02	943	0.96	0.60
Gamma	Conventional binary	1980	-	-	1980	1.03	973	1.00	2.03
	Stochastic-Prior	76	982	199	1257	153.6	1672	256.8	181.4
	Stochastic-PWM	76	678	110	864	1.8	1690	3.04	1.42

multiple bitstreams with small correlations. That results in significant area savings to the original implementation in [17]. A second LFSR was used for generating the Bernstein coefficients, making a total of two LFSRs and eight comparators to generate all the necessary bit streams in the "prior" approach.

2) The PWM Method: Here we discuss the process of synthesizing the gamma correction circuit using the PWM approach. The same process can be easily adapted to implement any other function that can be realized with the ReSC architecture.

Step 1. Frequency selection. At anytime only one input of the MUX is selected to be connected to the output. As a result, the PWM signals corresponding to the Bernstein coefficients can be generated with the same frequency. However, the circuit needs some level of independence between the six PWM signals corresponding to the inputs value of x. Fortunately, providing the required independence does not necessarily required generating signals with different frequencies, as was the case with multiplication. In the prior stochastic approach such independence could be provided by shifting the x streams for one or a few bits and so have a huge savings in the cost of SNG [34][35]. Similarly we can use a phase shift technique for the PWM approach to make independent copies of x. An additional step will select the best set of shift phases for the x signals that can lead to high quality outputs. Synthesis results showed a critical path of 0.60ns for the gamma correction circuit. Thus, accordingly, we chose 0.60ns as the period of the x signals and 0.9 as the period of the Bernstein coefficient signals. These periods are the scaled versions of (2ns, 3ns).

Step 2. Operation Time Determination. Since the gamma correction circuit is built on a MUX-based architecture, accurate outputs can be produced if the circuit runs for the LCM of the period of the inputs and the period of the PWM signals corresponding to the input x. Thus, the best operation time for the selected periods is their first common multiple or, 1.8ns. Note that using the phase shifting technique does not increase the operation time and highly accurate output can still be produced in LCM time by choosing the phases of the x signals appropriately.

Step 3. Clock Generation. Two clock generators are necessary for the Gamma correction circuit. One for generating a clock signal with a period of 0.9ns for the Bernstein PWM signals and another one for generating a clock signal with a period of 0.6ns. The latter drives the PWM generators responsible for generating the x signals. We used rings of 79 and 53 inverters to generate the required clock signals with periods of 0.9ns and 0.6ns, respectively.

Step 4. Phase Shift Calibration. A supplementary step is required to synthesize the ReSC architecture in the PWM approach. In the ReSC circuits, the results of adding independent copies of signal x determines which input of the MUX at any time must be connected to the output. Having six similar PWM signals, each signal can be shifted for a phase between 0 to the period of the signal. When using a ring of inverters as the clock generator, clock signals with the same frequency but different phases can be extracted from different stages of the ring. For the gamma correction circuit we needed six clock signals all with a fixed period of 0.6ns but each with a different phase. In several trials we measured the average error rates of processing one thousand random pixels when clock signals with different phases were extracted from different stages of the ring. For the final implementation, we chose the set of ring stages that led to the minimum average error rate.

3) Comparison: The pixels of the sample image were converted to their corresponding PWM signals and then processed by the implemented circuit. The mean of the error rates in processing all pixels of the sample image in the PWM approach was 2.18%, which is very close to the number reported for processing the sample image by the prior stochastic approach. The operation time for processing each image pixel has decreased from 153.6ns for the prior approach to only 1.8ns in the PWM approach. Also, the area-delay cost and energy consumptions are all significantly improved by the PWM approach when compared to the prior stochastic implementation. Note that we did not consider the cost of the required clock generator in the prior approach. If this cost were to be added, the improvement from the PWM approach would have been even greater.

Comparing the conventional binary implementation of the gamma correction function with the prior stochastic approach, we see that the latency of processing each image pixel, the energy dissipation, and the area-delay product of the stochastic approach are all significantly increased. The benefits of the prior stochastic approach are limited to around a 36 percent area saving and adding the ability to tolerate noise, which is an inherent property in SC. The PWM approach, on the other hand, not only inherits the noise tolerance advantage of the stochastic design, it also increases the area saving to 56 percent and brings the latency very close to the latency of the conventional binary design. Although the energy dissipation of the PWM approach is still more than that of the conventional design, it is much less than the energy dissipation of the prior stochastic approach.

V. Error Analysis

In this section we first define different sources of error in performing stochastic operations on PWM signals and then discuss the noise model and noise performance of the implemented PWM generator.

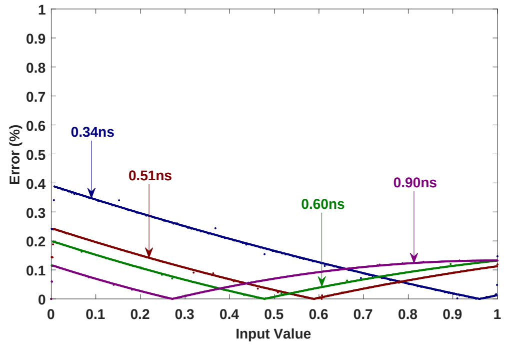

Fig. 19: Accuracy of the proposed PWM generator shown in Figure 5 in generating PWM signals with periods of 0.34ns, 0.51ns, 0.6ns, and 0.9ns.

A. Sources of Computational Error

There are five primary sources of error in performing stochastic operations on PWM signals:

1. E_G = error in generating the PWM signals.

A PWM generator has some inherent inaccuracies in converting real values to corresponding PWM duty cycles. This inaccuracy can be defined as the difference between the expected and the measured duty cycle in the generated signal.

$$E_G = |D - \frac{1}{T} \times T_{high}|$$

Figure 19 shows the accuracy of the PWM generator used in our simulations in converting real numbers in the [0, 1] interval to the corresponding PWM signals. As can be seen, the performance of the PWM generator is a function of the period of the signal and the duty cycle. For example, for small duty cycles, reducing the period lowers the accuracy of the generated signals. Note that in our simulations the error introduced in generating PWM signals was always less than 0.4%.

Additionally, achieving the desired frequency for the PWM signals is not always feasible, particularly when using ring oscillators as the clock generator. Changing the number of inverters is the simplest way to adjust the frequency of the oscillator. The oscillation period is twice the sum of the delay of all inverter gates, where the delay of one inverter gate in the selected 45nm library is 5.69ps. Considering that an odd number of inverter gates is required, the period can be increased (decreased) by adding (removing) an even number of inverters. Thus, the minimum change in period for this generator is 0.022ns. This limitation in controlling the period of the PWM generators can affect the accuracy of operations.

2. E_S = error due to skew noise.

For some stochastic operations, such as absolute value subtraction using XOR gates, perfectly synchronized PWM signals are necessary to produce accurate results. On-chip variations or other noise sources affecting ring oscillators can result in deviations from the expected period, phase shift or the slew rate of the signals. While these variations can affect the accuracy of the output signal, the results are still accurate to within the error bound expected for stochastic computation.

3. E_M = error in measuring output signals.

An analog integrator can be used to measure the fraction of the time the output signal is high. Longer rise and fall times and imperfect measurement of the high and low voltages (corresponding to digital “1” and “0” values) results in inaccuracies in measuring the correct output value. We compared the output values measured by our SPICE-level implementation of the integrator with the expected values from measuring the outputs produced by the Robert’s cross and Gamma circuit under ideal signal levels (HSPICE .ideal) when processing sample images. The average error rate of the measurements was 0.16% for the Robert’s cross and 0.12% for the Gamma correction circuit.

4. E_T = error due to truncation.

Truncation is another source of error in the PWM-based approach if the operation runs for any time other than the required operation time. For example, the multiplication operation must run the LCM or multiples of the LCM of the period of the PWM inputs to generate an accurate output. Running the operation for anytime less or more than the LCMs introduces truncation error.

5. E_A = error due to function approximation.

Functions implemented with SC typically must be approximated since a given function usually cannot be mapped directly to a stochastic operation. Our gamma correction operation, for example, used a Bernstein approximation of the exponential function. Prior work [17] has shown that a Bernstein approximation of degree of 6 is usually sufficient to reduce the average approximation error to below 0.1%

The overall error, E_{Total}, for the stochastic operations performed on PWM signals is bounded by the sum of the above error components:

$$E_{Total} = E_G + E_S + E_M + E_T + E_A$$

Considering the error rates we measured when processing the sample images using the synthesized Robert’s cross and

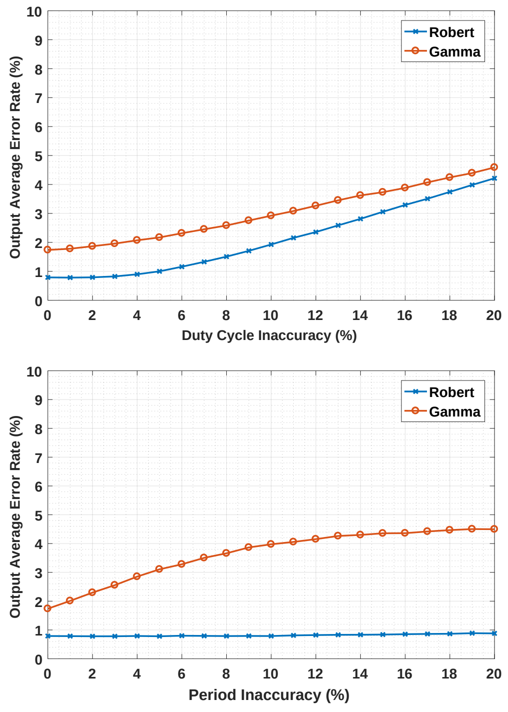

Fig. 20: Average error rate of the output images when processing the sample images using the proposed PWM-based approach for different rates of inaccuracy in the duty cycle (top) and in the period (bottom) of the PWM input signals. PWM signals are generated using an ideal PWM generator in HSPICE and the output signals are converted back to real values using an ideal integrator. Twenty trials were performed for each inaccuracy rate to ensure statistically significant results.

Gamma correction circuits with the PWM approach, some of these sources of errors can offset or compensate for each other, resulting in an acceptable total error. Note that, in an actual chip fabrication, the effect of thermal noise and the influence of process and temperature variations might introduce more inaccuracy in the generated signals which could produce higher error rates. Still, as Figure 20 shows, we expect that even if these fabrication sources of error introduce up to 20% relative error in the duty cycle and period of the PWM input signals, the stochastic circuits can still produce outputs with acceptably small errors.

B. Noise Modeling

Noise and linearity are definitely the most important concerns in analog circuits. In the following discussion, we analyze the noise contribution of each component in the PWM generator, and show that the proposed technique can satisfy the accuracy requirements even in the presence of thermal noise or process variations.

The ramp required for pulse width modulation is generated by charging a capacitor with a slope proportional to the input signal. If the input is coming from an image sensor, for instance, the output of the sensor is a current and can be directly integrated on a capacitor. On the other hand, there are cases like the coefficient inputs of the ReSC architecture where the input signal is a constant voltage and an active integrator such as Gm-C or R-OTA-C integrator must be used. We analyze these two cases separately.

Input source: Image sensor. In order to achieve 8-bit accuracy in PWM generation, the pulse width error must be less than $\frac{1}{2^9} \times T \approx 0.002T$, where T is the period of the PWM signal. There are two sources of error in the PWM generator:

1) Thermal noise:

a) Switched-capacitor noise: capacitors are inherently noiseless, but when they get switched, the thermal noise of the switch resistance accumulates on the capacitor, resulting in an equivalent rms noise voltage of KT/C [36]. This noise depends only on the capacitor size. Therefore, the maximum tolerable noise defines the minimum capacitance that can be used:

$$10\log_{10}\frac{\frac{0.5^2}{2}}{\frac{KT}{C}} \geq 8 \times 6.02 + 1.76 = 50\ dB$$

$$\frac{KT}{C} < \frac{0.5^2}{2}10^{-5} = 1.2510^{-6} \rightarrow C > 3.3\ fF$$

Since $C = 3.3\ fF$ was derived for room temperature, we choose $C = 5\ fF$ to allow some margin for temperature and process variations. This analysis shows a trade-off between capacitor area and circuit noise.

b) Comparator: the comparator is the key element in PWM generation. The comparator's resolution, i.e. the minimum voltage that causes a change in the output, determines the minimum detectable input current:

$$\text{The integration slope: } m_{LSB} = \frac{V_{res}}{t_{LSB}} = \frac{CMP_{res}}{0.002T} = \frac{i_{LSB}}{C}$$
$$\rightarrow i_{LSB} = \frac{CMP_{res}}{0.002T}$$

The comparator's resolution depends on the architecture. A typical comparator consists of a differential pair followed by a latch. The resolution of the comparator is given by $\frac{Vdd}{Comp_{gain}}$ where $Comp_{gain} = pre\text{-}amplifier_{gain} \times exp\ (\frac{t}{\tau})$. We show the $pre\text{-}amplifier_{gain}$ with Av. τ is the latch time constant measured by:

$$\tau = \frac{C_L \text{ (load capacitance at the output of the comparator)}}{G_m \text{ (transconductance of the cross-coupled latch)}}$$

The above equation shows that the comparator's resolution improves with time, i.e. one can achieve better resolution at the expense of longer delay [37], [38].

For 8-bit resolution with $1V\ Vdd$ for $1\,GHz$, the frequency$\rightarrow$ $Comp_{gain} > 512$, $t << 1ns$, $C_L = 1\ fF$, $Av = 16$ (easily achievable with low power/area), and t_d, or the maximum time that the comparator has to make a decision, is 0.001=1ps. Thus,

$$Av * exp(t \times 10^{15} \times G_m) = 512$$
$$\rightarrow exp(10^3 \times G_m) > 32 \rightarrow 10^3 * G_m > \ln(32) = 3.45$$
$$\rightarrow G_m > 3.45\,mA/v$$

Since we have high gain in the input stage, the noise of the latch does not matter (because the latch noise is divided by the input gain). The noise of the input transistors can result in pulse width variations, also known as jitter. A common formula for calculating jitter noise is [36]:

$$Jitter_{RMS} = \frac{Vnoise_{RMS}}{Slew\ rate}$$

Based on [39] we have:

$$Jitter_{RMS} = \frac{\sqrt{4KT\gamma/G_m} \times \sqrt{f}}{I/Cm}$$

It is worth noting that the effect of comparator noise on the PWM generator is the same as the ADC circuit presented in [6]. Also, note that process and temperature variation only affect the gain of the comparator, which can be taken into account during the design process. An example of a low-noise, low power comparator for 1 GHz frequency has been presented in [40].

2) Resetting speed: In each pulse generation cycle, the integrating capacitor must be discharged (reset) within the minimum time step, i.e. $\frac{T}{2^{N+1}}$. Therefore, the reset pulse width shrinks as the PWM frequency increases, imposing a limit on the maximum achievable speed. As calculated before, for 1ns period and 8-bit accuracy, t_{min}=2ps.

In summary, we have three sources of noise in the PWM generator: switched capacitor noise (KT/C), integrator noise, and comparator noise, where:

- KT/C is constant, because we change the current but the capacitor is fixed. For C=5 fF and room temperature, $\frac{KT}{C} = -57.81\,dB$
- The current has to scale linearly with speed, so the integrator noise decreases.
- Comparator noise results in jitter, so the impact increases with frequency. For $f = 1GHz$ it is $60\,dB$.
- Total distortion = integrator distortion (i.e. nonlinearity) = $-60\,dB$
- $SNDR = 10*\log_{10}(\frac{0.5\times V_{sig}^2}{Total_{noise}} + Total_{distortion}) = 6.02 * N + 1.76\ (dB)$
- For $Vdd = 1v \rightarrow 0.5 \times V_{sig}^2 = 0.5$
- For $f = 1GHz$, $Total_{noise} = 3 \times 10^{-6}$, so $SNDR = 51.5\,dB$ and $ENOB = 8.25$.

Input source: constant voltage. In case of voltage inputs, the transconductor (Gm cell) or the amplifier in the integrator also introduces noise, but the total noise is small and does not degrade the performance substantially.

VI. Conclusion

With a stochastic representation, computation has a pseudo *analog* character, operating on real-valued signals. This is certainly counterintuitive: why impose an analog view on digital values? Prior work has demonstrated that it is often advantageous to do so, both from the standpoint of the hardware resources required as well as the error tolerance of the computation. Many of the functions that we seek to implement for computational systems such as signal processing are *arithmetic* functions, consisting of operations like addition and multiplication. Complex functions, such as exponentials and trigonometric functions, are generally computed through polynomial approximations, so through multiplications and additions. Operations such as these can be implemented with remarkably simple hardware in the stochastic paradigm.

The cost incurred is to provide randomness. While randomness is never free, pseudo-randomness often suffices. The strategy proposed in this paper is to provide a form of pseudo-randomness through time-encoding of signals using pulse-width modulation (PWM). Such signals can be constructed with very common and inexpensive analog circuit structures. We have demonstrated that all the basic operations discussed in the literature on SC can be implemented on PWM signals.

Prior approaches to stochastic circuit design suffered from high run-time latency and correspondingly high energy use. Although the hardware cost of the core stochastic logic was negligible compared to the hardware cost of the conventional binary design, expensive stochastic number generators made them area and energy inefficient. With the proposed PWM approach, however, the latency, area and energy dissipation are all greatly reduced compared to the prior stochastic approaches. This new time-encoded approach inherits the fault tolerant advantage of stochastic design while working as fast and energy-efficiently as the conventional binary design. Fault tolerant capability, a lower hardware cost and a smaller area-delay product make the proposed PWM approach a better choice than the conventional binary design.

Acknowledgment

This work was supported in part by National Science Foundation grant no. CCF-1408123. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. The authors would like to thank Prof. Mahdi Asgari of Oklahoma State University for helpful discussions on relatively prime periods and the proof provided in Section III.A, and also the anonymous reviewers whose comments and suggestions have helped to improve the quality of the manuscript.

References

[1] Armin Alaghi and John P. Hayes. Survey of stochastic computing. *ACM Trans. Embed. Comput. Syst.*, 12(2s):92:1–92:19, May 2013.

[2] J.P. Hayes. Introduction to stochastic computing and its challenges. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pages 1–3, June 2015.

[3] B.R. Gaines. Stochastic computing systems. In JuliusT. Tou, editor, *Advances in Information Systems Science*, Advances in Information Systems Science, pages 37–172. Springer US, 1969.

[4] Peng Li, D.J. Lilja, Weikang Qian, K. Bazargan, and M.D. Riedel. Computation on stochastic bit streams digital image processing case studies. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(3):449–462, March 2014.

[5] M. H. Najafi and M. E. Salehi. A Fast Fault-Tolerant Architecture for Sauvola Local Image Thresholding Algorithm Using Stochastic Computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(2):808–812, Feb 2016.

[6] A. Alaghi, Cheng Li, and J.P. Hayes. Stochastic circuits for real-time image-processing applications. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–6, May 2013.

[7] N. Onizawa, D. Katagiri, K. Matsumiya, W. J. Gross, and T. Hanyu. Gabor filter based on stochastic computation. *IEEE Signal Processing Letters*, 22(9):1224–1228, Sept 2015.

[8] D. Fick, G. Kim, A. Wang, D. Blaauw, and D. Sylvester. Mixed-signal stochastic computation demonstrated in an image sensor with integrated 2d edge detection and noise filtering. In *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*, pages 1–4, Sept 2014.

[9] S.S. Tehrani, W.J. Gross, and S. Mannor. Stochastic decoding of ldpc codes. *Communications Letters, IEEE*, 10(10):716–718, Oct 2006.

[10] X. R. Lee, C. L. Chen, H. C. Chang, and C. Y. Lee. A 7.92 gb/s 437.2 mw stochastic ldpc decoder chip for ieee 802.15.3c applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(2):507–516, Feb 2015.

[11] Naoya Onizawa, Warren J. Gross, Takahiro Hanyu, and Vincent C. Gaudet. Asynchronous stochastic decoding of ldpc codes: Algorithm and simulation model. *IEICE Transactions on Information and Systems*, 97(9):2286–2295, 2014.

[12] B.D. Brown and H.C. Card. Stochastic neural computation. i. computational elements. *Computers, IEEE Transactions on*, 50(9):891–905, Sep 2001.

[13] Kyounghoon Kim, Jungki Kim, Joonsang Yu, Jungwoo Seo, Jongeun Lee, and Kiyoung Choi. Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 124:1–124:6, New York, NY, USA, 2016. ACM.

[14] Bingzhe Li, M. Hassan Najafi, and David J. Lilja. Using stochastic computing to reduce the hardware requirements for a restricted boltzmann machine classifier. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, pages 36–41, New York, NY, USA, 2016. ACM.

[15] Yuan Ji, Feng Ran, Cong Ma, and David J. Lilja. A hardware implementation of a radial basis function neural network using stochastic logic. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, pages 880–883, San Jose, CA, USA, 2015. EDA Consortium.

[16] Y. Liu, H. Venkataraman, Z. Zhang, and K. K. Parhi. Machine learning classifiers using stochastic logic. In *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pages 408–411, Oct 2016.

[17] Weikang Qian, Xin Li, M.D. Riedel, K. Bazargan, and D.J. Lilja. An architecture for fault-tolerant computation with stochastic logic. *Computers, IEEE Transactions on*, 60(1):93–105, Jan 2011.

[18] Qianying Tang, Bongjin Kim, Yingjie Lao, K.K. Parhi, and C.H. Kim. True random number generator circuits based on single- and multi-phase beat frequency detection. In *Custom Integrated Circuits Conference (CICC), 2014 IEEE Proceedings of the*, pages 1–4, Sept 2014.

[19] Won Ho Choi, L.V. Yang, Jongyeon Kim, A. Deshpande, Gyuseong Kang, Jian-Ping Wang, and C.H. Kim. A magnetic tunnel junction based true random number generator with conditional perturb and real-time output probability tracking. In *Electron Devices Meeting (IEDM), 2014 IEEE International*, pages 12.5.1–12.5.4, Dec 2014.

[20] B. Moons and M. Verhelst. Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 4(4):475–486, Dec 2014.

[21] M. H. Najafi, D. J. Lilja, M. Riedel, and K. Bazargan. Polysynchronous stochastic circuits. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 492–498, Jan 2016.

[22] Solomon W. Golomb and Guang Gong. Signal design for good correlation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2004.

[23] K. Kim, J. Lee, and K. Choi. An energy-efficient random number generator for stochastic circuits. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 256–261, Jan 2016.

[24] B. Murmann. "ADC Performance Survey 1997-2015," [online]. Available: http://web.stanford.edu/ murmann/adcsurvey.html, 2015.
[25] Xueqin Lu, Shuguo Chen, Chenning Wu, and Mingzhu Li. The pulse width modulation and its use in induction motor speed control. In *Computational Intelligence and Design (ISCID), 2011 Fourth International Symposium on*, volume 2, pages 195–198, Oct 2011.
[26] S. L. Toral, J. M. Quero, and L. G. Franquelo. Stochastic pulse coded arithmetic. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 1, pages 599–602 vol.1, 2000.
[27] J. M. Quero, C. L. Janer, J. G. Ortega, and L. G. Franquelo. D/A Converter ASIC Uses Stochastic Logic. In *EDN*, pages 86–88, Oct. 1996.
[28] W.J. Poppelbaum, A. Dollas, J.B. Glickman, and C. O'Toole. Unary processing. In *Advances in Computers*, volume 26, pages 47 – 92. Elsevier, 1987.
[29] P. Mars and W.J. Poppelbaum. Stochastic and Deterministic Averaging Processors. IEE digital electronics and computing series. The institution of Electrical Engineers, 1981.
[30] MATLAB. *version 9.0.0 (R2016a)*. The MathWorks Inc., Natick, Massachusetts, 2016.
[31] Armin Alaghi and John P. Hayes. On the functions realized by stochastic computing circuits. In *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, GLSVLSI '15, pages 331–336, New York, NY, USA, 2015. ACM.
[32] M. H. Najafi and D. J. Lilja. High-Speed Stochastic Circuits Using Synchronous Analog Pulses. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2017.
[33] Synopsys. *Design Compiler Users Manual*. http://www.synopsys.com/.
[34] H. Ichihara, S. Ishii, D. Sunamori, T. Iwagaki, and T. Inoue. Compact and accurate stochastic circuits with shared random number sources. In *Computer Design (ICCD), 2014 32nd IEEE International Conference on*, pages 361–366, Oct 2014.
[35] Zhiheng Wang, Naman Saraf, Kia Bazargan, and Arnd Scheel. Randomness meets feedback: Stochastic implementation of logistic map dynamical system. In *Design Automation Conference (DAC)*, 2015.
[36] D. JOHNS and K. MARTIN. Analog integrated circuit design. 1997.
[37] B. Razavi. The cross-coupled pair - part i [a circuit for all seasons]. *IEEE Solid-State Circuits Magazine*, 6(3):7–10, Summer 2014.
[38] B. Razavi. The cross-coupled pair - part ii [a circuit for all seasons]. *IEEE Solid-State Circuits Magazine*, 6(4):9–12, Fall 2014.
[39] T. Sepke, P. Holloway, C. G. Sodini, and H. S. Lee. Noise analysis for comparator-based circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(3):541–553, March 2009.
[40] Masaya Miyahara, Yusuke Asada, Daehwa Paik, and Akira Matsuzawa. A low-noise self-calibrating dynamic comparator for high-speed adcs. In *Solid-State Circuits Conference, 2008. A-SSCC '08. IEEE Asian*, pages 269–272, Nov 2008.

M. Hassan Najafi received the B.Sc. degree in computer engineering from University of Isfahan, Iran, and the M.Sc. degree in computer architecture from University of Tehran, Iran, in 2011 and 2014, respectively. He is currently working toward the Ph.D. degree as a research assistant at ARCTiC Labs in the Department of Electrical and Computer Engineering, University of Minnesota, Twin cities. His research interests include stochastic and approximate computing, computer-aided design of integrated circuits, low power design, and designing fault tolerant systems.

Shiva Jamali-Zavareh received her B.Sc. degree in electrical engineering from University of Tehran, Tehran, Iran, in 2011, and her M.Sc. degree in microelectronic circuit design from Aalto University, Espoo, Finland, in 2014. She is currently working toward her Ph.D. degree in analog design lab in the Department of Electrical and Computer Engineering, University of Minnesota, Twin cities. Her research interests include analog front-ends, data converters, RF circuit design, and stochastic computing.

David J. Lilja (F06) received the B.S. degree in computer engineering from Iowa State University in Ames, IA, USA, and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign in Urbana, IL, USA. He is currently the Schnell Professor of Electrical and Computer Engineering at the University of Minnesota in Minneapolis, MN, USA, where he also serves as a member of the graduate faculties in Computer Science, Scientific Computation, and Data Science. Previously, he served ten years as the head of the ECE department at the University of Minnesota, and worked as a research assistant at the Center for Supercomputing Research and Development at the University of Illinois, and as a development engineer at Tandem Computers Incorporated in Cupertino, California. He was elected a Fellow of the Institute of Electrical and Electronics Engineers (IEEE) and a Fellow of the American Association for the Advancement of Science (AAAS).

Marc D. Riedel (SM12) received the B.Eng. degree in electrical engineering from McGill University, Montreal, QC, Canada, and the M.Sc. and Ph.D. degrees in electrical engineering from the California Institute of Technology (Caltech), Pasadena, CA, USA. He is currently an Associate Professor of electrical and computer engineering with the University of Minnesota, Minneapolis, MN, USA, where he is a member of the Graduate Faculty of biomedical informatics and computational biology. From 2004 to 2005, he was a Lecturer of computation and neural systems with Caltech. He was with Marconi Canada, CAE Electronics, Toshiba, and Fujitsu Research Labs. Dr. Riedel was a recipient of the Charl H. Wilts Prize for the Best Doctoral Research in Electrical Engineering at Caltech, the Best Paper Award at the Design Automation Conference, and the U.S. National Science Foundation CAREER Award.

Kia Bazargan (SM07) received the B.Sc. degree in computer science from Sharif University, Tehran, Iran, and the M.S. and Ph.D. degrees in electrical and computer engineering from Northwestern University, Evanston, IL, USA, in 1998 and 2000, respectively. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA. Dr. Bazargan was a recipient of the US National Science Foundation Career Award in 2004. He was a Guest Co-Editor of the ACM Transactions on Embedded Computing Systems Special Issue on Dynamically Adaptable Embedded Systems in 2003. He was on the technical program committee of a number of the IEEE/ACM-sponsored conferences, including Field Programmable Gate Array, Field Programmable Logic, Design Automation Conference (DAC), International Conference on Computer-Aided Design, and Asia and South Pacific DAC. He was an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 2005 to 2012. He is a Senior Member of the IEEE Computer Society.

Ramesh Harjani (F05) is the Edgar F. Johnson Professor in the Department of Electrical and Computer Engineering at the University of Minnesota. He received his Ph.D. degree from Carnegie Mellon University in 1989, his M.S. degree from the Indian Institute of Technology, New Delhi, in 1984, and his B.S. degree from the Birla Institute of Technology and Science, Pilani, in 1982, all in electrical engineering. Prior to joining the University of Minnesota, he was with Mentor Graphics Corp. in San Jose, California. He co-founded Bermai, Inc, a startup company developing CMOS chips for wireless multimedia applications in 2001. He has been a visiting professor at Lucent Bell Labs, Allentown, Pennsylvania, and the Army Research Labs, Adelphi, Maryland. His research interests include analog/RF circuits for wired and wireless communications.